Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol.7, No.9, 2016
www.iiste.org

Fast Overflow Detection Scheme by Operands Examinations Method for Length Three Moduli Sets

M. I. Daabo[1*] K. A. Gbolagade[2] P.A. Agbdemnab[3]
1.Department of Computer Science, Faculty of Mathematical Sciences, University for Development Studies Navrongo, Ghana
2.Department of Computer, Library and Information Science, College of Information and Communication Technology, Kwara State University, Nigeria
3.Department of Computer Science, Faculty of Mathematical Sciences, University for Development Studies Navrongo, Ghana

Abstract
In this paper, we present a fast overflow detection scheme by Operands Examination Method (OEM) for 3-Moduli Sets. The method examines the sum of the Mixed Radix Digits (MRDs) computed using the Mixed Radix Conversion (MRC) method to detect overflow for the sum of operands. It is observed that by reducing larger numbers into smaller numbers, the OEM approach makes computations easier and faster. The proposed scheme is further implemented on the Moduli Set $\{2^n - 1, 2^n, 2^{2n+1} - 1\}$ for validation purposes. Theoretically, the scheme proves to be more efficient in detecting overflow as compared to current existing overflow detection schemes.
Keywords: Residue Number System, Operands Examination Method, Overflow Detection, Mixed Radix Digits, Mixed Radix Conversion.

1. INTRODUCTION

The problem of overflow is one of the major challenges that limit the full implementation of Residue Number System (RNS) Architecture in general purpose computing. Overflow is said to occur in RNS computations when a calculation produces results greater than the valid dynamic range of the given RNS. Thus overflow is an error that usually occur during addition and multiplication dominated operations where larger numbers are involved. If overflow is not detected properly, then the whole RNS Architecture may fail as wrong results may be used for further computations.

RNS is a Non Weighted Number System that makes computations easier and faster because of its inherent features of digit to digit computations, fault tolerance, parallelism, high computational speed and low power dissipation. The system is applied in the fields of Digital Signal Processing (DSP) intensive computations like digital filtering, convolutions, correlations, Discrete Fourier Transform (DFT) computations, Fast Fourier Transform (FFT) computations and direct digital frequency synthesis (Omondi & Premkumar, 2007), (Gbolagade, 2013) and (Gbolagade et al., 2010).

RNS is determined by a set *S,* of *N* integers that are pair-wise relatively prime. That is $S = \{m_1, m_2, \ldots, m_N\}$ Where $gcd(m_i, m_j) = 1$ for $i, j = 1, \ldots, N$ and $i \neq j$, and gcd means the greatest common divisor [7]. Every integer X in $[0, M-1]$ can be uniquely represented with a *N*-tuple where, $M = \prod_{i=1}^{N} m_i$, $X \rightarrow (x_1, x_2, \ldots, x_N)$ and $x_i = |X|_{m_i} = (X \bmod m_i)$; for $i = 1$ to N. The set *S* and the number x_i are called the moduli set and residue of *X* modulo m_i respectively.

There are a number of techniques of converting an RNS number to its binary equivalent form, famous among which are the Chinese Remainder Theorem (CRT) and the Mixed Radix Conversion (MRC). Any other technique is basically a modification of these two techniques. The MRC is calculated according to equations (1) and (2).

In general, during the addition of two RNS numbers, overflow can be detected when the result of the addition is less than one of the addends. For example, given two RNS numbers X and Y such that $Z = X + Y \bmod M$ where $X \geq 0$ and $Y < M$, overflow will only occur when $Z < X$.

Another efficient way to detect overflow in RNS is via parity checking (Omondi & Premkumar, 2007) and (Rouhifar et al., 2011). It indicates whether an integer is even or odd. Suppose two integers (X, Y) have the same parity: $Z = X + Y$. An overflow occurs if Z is odd. Contrary, if (X, Y) have different parity, then an overflow occurs if Z is even. This technique is one of the best and fastest suggested methods to detect the overflow in RNS. However, this technique is only suitable for moduli sets with odd Dynamic Range (DR). But RNS systems that have even DR have more attractive features than those with odd DR. This is because using (2^n) modulo which tends dynamic ranges to even, greatly simplifies and reduces the delay and complexity of the scheme (Younes & Steffan, 2013). Thus the need to devise techniques of detecting overflow in moduli sets with

even dynamic range.

In recent years, researchers have made considerable efforts to design overflow detection schemes to handle both odd and even dynamic ranges schemes which do not rely on the traditional techniques such as CRT and Mixed Radix Conversion (MRC) for full reverse conversion; these proposed RNS overflow detection algorithms still rely on the later (Younes & Steffan, 2013) and other costly and time consuming procedures such as base extension, group number and sign detection as in (Rouhifar et al., 2011), (Molahosseini & Navi, 2007) and (Hosseinzadeh et al., 2009). The scheme in (Younes & Steffan, 2013) is said to be universal since it is able to detect overflow in both signed and unsigned RNS representation but require high hardware resources since it relies on full reverse converters and comparators.

In this paper, a fast overflow detection scheme by Operands Examination Method for 3-Moduli Sets is presented. The method examines the sum of the Mixed Radix Digits (MRDs) computed using the Mixed Radix Conversion (MRC) method to detect overflow for the sum of operands without having to do full reverse conversion.

The rest of the paper is organized as follows: Section 2 presents the proposed algorithms. In Section 3, an implementation of the proposed scheme is presented, with numerical examples. The performance of the proposed scheme is evaluated in Section 4 whiles the paper is concluded in Section 5.

2. PROPOSED ALGORITHMS

For a given RNS number, the MRC shown in Equation (1), represents only the equivalent decimal number which is within the legitimate range $[0, M-1]$. Numbers and their sums which fall within the legitimate range are said to be valid numbers and do not show overflow condition. However, numbers and their sums which fall outside this range are said be in the overflow region and must be investigated. For instance if we choose two decimal numbers X and Y, then their respective RNS numbers can be represented by (x_1, x_2, x_3) and (y_1, y_2, y_3). The MRDs (a_1, a_2, a_3), (b_1, b_2, b_3), and (ρ_1, ρ_2, ρ_3) can be computed to respectively represent X, Y and Z. In this case Z is the sum of X and Y. The MR is given as

$$X = a_1 + a_2 m_1 + a_3 m_1 m_2 + \cdots + m_1 m_2 m_3 \ldots m_{n-1} \quad (1)$$

Where the MRDs, $a_i, i = 1, n$ can be computed as:

$$a_1 = x_1$$
$$a_2 = |(x_2 - a_1)|m_1^{-1}|_{m_2}|_{m_2}$$
$$a_3 = \left|((x_3 - a_1)|m_1^{-1}|_{m_3} - a_2)|m_2^{-1}|_{m_3}\right|_{m_3}$$
$$\vdots$$
$$a_n = \left|((\ldots(x_n - a_1)|m_1^{-1}|_{m_n} - a_2)|m_2^{-1}|_{m_3} - \cdots - a_{n-1})|m_{n-1}^{-1}|_{m_n}\right|_{m_n} \quad (2)$$

Given the MRD $a_i, 0 \leq a_i < m_i$, any positive number in the interval $[0, \prod_{i=1}^{n} m_i - 1]$ can be uniquely represented in equation (1).

2.1 Determination of Overflow Detection Algorithm for a 3- Moduli Set

From Equation (1), we can represent X and Y as:

$$X = a_1 + a_2 m_1 + a_3 m_1 m_2 \quad (3)$$
$$Y = b_1 + b_2 m_1 + b_3 m_1 m_2 \quad (4)$$

From Equation (3) and (4) the sum of X and Y is given as

Z = X+Y

$$Z = (a_1 + a_2 m_1 + a_3 m_1 m_2) + (b_1 + b_2 m_1 + b_3 m_1 m_2)$$
$$Z = (a_2 + b_1) + (a_2 + b_2) m_1 + (a_3 + b_3) m_1 m_2$$
$$Z = (\rho_1) + (\rho_2) m_1 + (\rho_3) m_1 m_2 \quad (5)$$

Therefore $Z = X + Y$ and $\rho_i = a_i + b_i, \; i = 1 \; to 3$.

Property 1: Equation (1) only represents valid numbers within the legitimate range $[0, M-1]$ for a given RNS. Thus numbers outside the dynamic range are in the overflow region.

Theorem 1: Overflow occurs in the sum of X and Y if the following hold true;

$$\rho_3 > m_3 - 1$$
$$\rho_3 = m_3 - 1 \text{ AND } \rho_2 > m_2 - 1$$

$$\rho_3 = m_3 - 1 \text{ AND } \rho_2 = m_2 - 1 \text{ AND } \rho_1 > m_1 - 1 \quad (6)$$

Proof:

If we assume Equation (6) is true;

Then from Equation (6) ρ_n has weight $w_n = \prod_{i=1}^{n-1} m_i$ which implies the value of Z can be expressed as;

$$Z \geq \rho_1 + \rho_2 m_1 + \rho_3 m_1 m_2 \quad (7)$$

If we substitute $\rho_3 = m_3 - 1$ AND $\rho_2 = m_2 - 1$ AND $\rho_1 = m_1 - 1$ in Equation (7), then we have;

$$Z \geq m_1 - 1 + (m_2 - 1)m_1 + (m_3 - 1)m_1 m_2$$
$$Z \geq m_1 - 1 + m_1 m_2 - m_1 + m_1 m_2 m_3 - m_1 m_2$$
$$Z \geq m_1 m_2 m_3 - 1$$

If $Z = m_1 m_2 m_3 - 1$, then Z lies within the legitimate range and there is no overflow.

Thus from property (1), Z will only lie outside the legitimate range $[0, M-1]$ whenever Equation (6) holds true. And which means that overflow will occur in the computation of Z.

2.2 The Algorithms for a Length Three Moduli Set

In this section, we propose overflow detection algorithms for a three moduli set. Given any two RNS numbers $X = (x_1, x_2, x_3)$ and $Y = (y_1, y_2, y_3)$, the following operations can be performed to detect overflow in the addition of X and Y:

I. Compute the MRDs (a_1, a_2, a_3) and (b_1, b_2, b_3) of X and Y.

II. Compute (ρ_1, ρ_2, ρ_3), where $\rho_i = a_i + b_i$ for $i = 1 \text{ to } 3$.

III. From Theorem 1, overflow occurs in $X + Y$ if the following hold true:

$$\rho_3 > m_3 - 1.$$
$$\rho_3 = m_n - 1 \text{ AND } \rho_{n-1} > m_{n-1} - 1$$
$$\rho_3 = m_3 - 1 \text{ AND } \rho_2 = m_2 - 1 \text{ AND } \rho_3 > m_3 - 1$$

2.3 Overflow Detection Algorithms for the moduli Set $\{2^n-1, 2^n, 2^{2n+1}-1\}$

This section presents a simplified overflow detection scheme for the moduli set $\{2^n - 1, 2^n, 2^{2n+1} - 1\}$ using the OEM algorithm above. This is a three moduli set with a larger dynamic range. Given the RNS numbers $X = (x_1, x_2, x_3)$ and $Y = (y_1, y_2, y_3)$ with respect to the moduli set $\{2^n - 1, 2^n, 2^{2n+1} - 1\}$ then the simplified algorithms for the given moduli set is given as;

I. Compute the MRDs (a_1, a_2, a_3) and (b_1, b_2, b_3) of X and Y respectively.

II. Compute (ρ_1, ρ_2, ρ_3), where $\rho_i = a_i + b_i$ for $i = 1 \text{ to } 3$.

III. From Theorem 1, overflow will occur in $X + Y$ if the following hold true:

$$\rho_3 > 2^n - 2$$
$$\rho_3 = 2^n - 2 \; AND \; \rho_2 > 2^n - 1$$
$$\rho_3 = 2^n - 2 \; AND \; \rho_2 = 2^n - 1 \; AND \; \rho_1 > 2^{2n+1} - 2.$$

Theorem 2: Given the moduli set $\{2^n - 1, 2^n, 2^{2n+1} - 1\}$, where $m_1 = 2^{2n+1} - 1$, $m_2 = 2^n$ and $m_3 = 2^n - 1$ for every integer $n > 1$, the following hold true:

$$|m_1^{-1}|_{m_2} = -1 \quad (8)$$
$$|m_2^{-1}|_{m_3} = 1 \quad (9)$$
$$|m_1^{-1}|_{m_3} = 1 \quad (10)$$

Proof: It can be demonstrated that if $|(2^{2n+1} - 1)(-1)|_{2^n} = 1$, then -1 is the multiplicative inverse of m_1 with respect to m_2.

This implies $|(2^{2n+1} - 1)(-1)|_{2^n} = |(-1)(-1)|_{2^n} = |1|_{2^n} = 1$

Thus (8) holds true. Also, if it can be demonstrated that $|(2^n) \times (1)|_{2^n - 1} = 1$, then 1 is the multiplicative inverse of m_2 with respect to m_3.

i.e. $|(2^n) \times (1)|_{2^n - 1} = |(2^n - 1 + 1) \times (1)|_{2^n - 1} = |(1) \times (1)|_{2^n - 1} = |1|_{2^n - 1} = 1$

Thus equation (9) holds true. In the same way, if it can be demonstrated that $|(2^{2n+1}-1)\times(1)|_{2^n-1}=1$, then 1 is the multiplicative inverse of m_1 with respect to m_3.
Thus we show that
$|(2^{2n+1}-1)\times(1)|_{2^n-1} = |2(2^{2n})-1)\times(1)|_{2^n-1}$
$= |(2-1)\times(1)|_{2^n-1} = |1|_{2^n-1} = 1$
Thus, Equation)10(holds true.
Therefore we can re-write (2) as;
$a_1 = x_1$
$a_2 = |(x_2-a_1)(-1)|_{2^n} = |x_1-x_2|_{2^n}$
$a_3 = |((x_3-a_1)1-a_2)1|_{2^n-1} = |(x_3-a_1)-a_2|_{2^n-1}$ (11)

3. IMPLEMENTATION

The proposed scheme uses basic and smaller logic units when compared to similar existing designs. The major goal in this case is to compute ρ_1, ρ_2, and ρ_3 which are to be transmitted for post processing i.e. $\rho_i = a_i + b_i$.
Thus;
$\rho_1 = a_1 + b_1 = x_1 + y_1$ (12)
since $a_1 = x_1$ and $b_1 = y_1$.
Similarly; $\rho_2 = a_2 + b_2$ (13)
Where $a_2 = |x_1 - x_2|_{2^n} = ||x_1|_{2^n} - |x_2|_{2^n}|_{2^n} = |x_1 - |x_2|_{2^n}|_{2^n}$
And $b_2 = |y_1 - y_2|_{2^n} = ||y_1|_{2^n} - |y_2|_{2^n}|_{2^n} = |y_1 - |y_2|_{2^n}|_{2^n}$ (14)
It can be seen that x_2 and y_2 are already modulo 2^n numbers and hence requires less hardware.
Finally $\rho_3 = a_3 + b_3$ (15)
where $a_3 = |z_1 - a_2|_{2^n-1}$ and $b_3 = |z_2 - b_2|_{2^n-1}$
$z_1 = |(x_3 - x_1)|_{2^n-1}$ and $z_2 = |(y_3 - y_1)|_{2^n-1}$

3.1 Numerical Illustrations

In this sub-section, we present numerical illustrations of the proposed scheme. If we let n = 2 in the moduli set $\{2^n-1, 2^n, 2^{2n+1}-1\}$ then we have $\{31, 4, 3\}$.

Checking overflow in the sum of 180 and 300.

$180 = (25,0,0)_{RNS\langle 31|4|3\rangle}$
$= (11001,00,00)_{RNS\langle 11111|100|11\rangle}$
$300 = (21,0,0)_{RNS\langle 31|4|3\rangle}$
$= (10101,00,00)_{RNS\langle 11111|100|11\rangle}$
Implies $(25,0,0)_{RNS\langle 31|4|3\rangle} + (21,0,0)_{RNS\langle 31|4|3\rangle} = ((11001,00,00) + (10101,00,00))_{RNS\langle 11111|100|11\rangle}$
$= (00110,00,00)_{RNS\langle 11111|100|11\rangle}$
RNS to decimal conversion of $(00110,00,00)_{RNS\langle 11111|100|11\rangle}$ results in the decimal number 108 which indicates that the sum of 180 and 300 does not result in overflow in the RNS system. But it is well known that the sum of the decimal numbers 180 and 300 is 480 which gives a clear sign that overflow has occurred.

Checking for RNS overflow using the proposed algorithm

$\rho_1 = a_1 + b_1 = x_1 + y_1 = 11001 + 10101 = 101110$
$\rho_2 = a_2 + b_2 = |x_1 - |x_2|_{2^n}|_{2^n} + |y_1 - |y_2|_{2^n}|_{2^n}$
$= |11001 - |00|_{100}|_{100} + |10101 - |00|_{100}|_{100}$
$= |01|_{100} + |01|_{100} = 01 + 01 = 10$
$\rho_3 = a_3 + b_3$
$= |z_1 - a_2|_{2^n-1} + |z_2 - b_2|_{2^n-1}$
$= ||010|_{11} - |01|_{11}|_{11} + ||000|_{11} - |01|_{11}|_{11}$
$= |001|_{11} + |10|_{11} = 01 + 10 = 11$.

Post Processing

From Theorem 1, overflow will occur if the following hold true:
$\rho_3 > 10$
$\rho_3 = 10\ AND\ \rho_2 > 11$
$\rho_3 = 10\ AND\ \rho_2 = 11\ AND\ \rho_1 > 11110$

A close examination of the values at post processing shows that overflow has occurred since $\rho_3 = 11$ which is greater than 10.

Checking for overflow in the sum of 5 and 21.

$5 = (5,1,2)_{RNS\langle 31|4|3\rangle} = (00101,01,10)_{RNS\langle 11111|100|11\rangle}$
$21 = (21,1,0)_{RNS\langle 31|4|3\rangle} = (10101,01,00)_{RNS\langle 11111|100|11\rangle}$

Implies

$$(5,1,2)_{RNS\langle 31|4|3\rangle} + (21,1,0)_{RNS\langle 31|4|3\rangle} = ((00101,01,10) + (10101,01,00))_{RNS\langle 11111|100|11\rangle}$$
$$= (11010,10,10)_{RNS\langle 11111|100|11\rangle}$$

RNS to decimal conversion of $(11010,10,10)_{RNS\langle 11111|100|11\rangle}$ results in the decimal number 26 which is the correct result of $5 + 21$. This does not indicate any sign of overflow in the RNS system.

Checking for RNS overflow using the proposed algorithm

$\rho_1 = a_1 + b_1 = x_1 + y_1 = 00101 + 10101 = 11010$
$\rho_2 = a_2 + b_2 = |x_1 - |x_2|_{2^n}|_{2^n} + |y_1 - |y_2|_{2^n}|_{2^n}$
$= |00101 - |01|_{100}|_{100} + |10101 - |01|_{100}|_{100}$
$= |00|_{100} + |00|_{100} = 00 + 00 = 00$
$\rho_3 = a_3 + b_3 = a_3$
$= |z_1 - a_2|_{2^n-1} + |z_2 - b_2|_{2^n-1}$
$= ||000|_{11} - |00|_{11}|_{11} + ||000|_{11} - |00|_{11}|_{11}$
$= |000|_{11} + |00|_{11} = 00 + 00 = 00$

Post Processing

From Theorem 1, overflow will occur if the following hold true:
$\rho_3 > 10$
$\rho_3 = 10\ AND\ \rho_2 > 11$
$\rho_3 = 10\ AND\ \rho_2 = 11\ AND\ \rho_1 > 11110$

In examining the values of ρ_3 at post processing, it reveals that overflow will not occur since $\rho_3 = 00$ is less than 10. The proposed scheme is therefore capable of detecting overflow in the given moduli set.

4. PERFORMANCE EVALUATION

To appropriately evaluate the performance of the proposed scheme, we need to compare it with similar best known existing schemes in terms of delay and area cost. Our proposal eliminates the computation of the modulo M sum of X and Y and uses smaller modulo operations. In addition, our proposed technique is involved in the manipulation of smaller numbers compared to other techniques. Generally, it is known that the smaller the numbers involved in arithmetic computations, the faster the processing time. Finally, the scheme is capable of converting the sum of any two RNS numbers in to its decimal equivalent. Thus, the proposed scheme outperforms other existing schemes in terms of delay and area cost.

5. CONCLUSION

In this paper, RNS overflow detection scheme for addition has been proposed for length three moduli sets. The proposed scheme is implemented on the moduli set $\{2^n - 1, 2^n, 2^{2n+1} - 1\}$. Numerical illustrations were carried out on the scheme and theoretically, the scheme performance is better than current existing similar designs in terms of both area cost and delay.

6. REFERENCES

Omondi, A. & Premkumar, B. (2007), 'Residue Number Systems: Theory and Implementation', *Imperial College Press. UK.*

Gbolagade, K. A. (2013), 'An Efficient MRC based RNS-to-Binary Converter for the $\{2^{2n} - 1, 2^n, 2^{2n+1} - 1\}$ Moduli Set', *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) 2(4).*

Gbolagade, K. A., Chaves, R., Sousa, L. & Cotofana S.D. (2010), 'An improved reverse converter for the Moduli set', *IEEE International Symposium on Circuits and Systems (ISCAS 2010), 2103-2106, Paris, France.*

Younes, D. & Steffan, P. (2013), 'Universal approaches for overflow and sign detection in residue number system based on $\{2^n-1, 2^n, 2^n+1\}'$, *The Eighth International Conference on Systems (ICONS 2013), 77 – 84.*

Rouhifar, M., Hosseinzadeh, M., Bahanfar, S. & Teshnehlab, M. (2011), 'Fast Overflow Detection in Moduli set $\{2^n-1, 2^n, 2^n+1\}'$, *International Journal of Computer Science Issues, 8(3), 407-414.*

Hosseinzadeh, M., Molahosseini, A.S. & Navi, K. (2009), 'A parallel Implementation of the Reverse Converter for the moduli set $\{2^n-1, 2^n, 2^{n-1}-1\}'$, *World Academy of Science, Engineering and Technology, Vol. 55, 494-498.*

Molahosseini, A,S, & Navi, K. (2007), 'New Arithmetic residue to binary Converters', *International Journal of Computer Sciences and Engineering Systems, 1(4), 295-299.*